

VU Research Portal

How to Keep Your Memory Safe and Your Software Fast

Kroes, T.

2020

document version

Publisher's PDF, also known as Version of record

[Link to publication in VU Research Portal](#)

citation for published version (APA)

Kroes, T. (2020). *How to Keep Your Memory Safe and Your Software Fast*. [PhD-Thesis - Research and graduation internal, Vrije Universiteit Amsterdam].

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:

vuresearchportal.ub@vu.nl

VRIJE UNIVERSITEIT

**HOW TO KEEP YOUR MEMORY SAFE
AND YOUR SOFTWARE FAST**

PH.D. THESIS

Taddeüs Kroes

The research reported in this dissertation was conducted at the Faculty of Science, at the Department of Computer Science, of the Vrije Universiteit Amsterdam.

This work is part of the research programme *Vici* with project number 639.023.309 titled “Dowsing”, which is funded by the Dutch Research Council (NWO).

Copyright © 2020 by Taddeüs Kroes

Cover and chapter illustrations by Mei-Li Nieuwland

VRIJE UNIVERSITEIT

**HOW TO KEEP YOUR MEMORY SAFE
AND YOUR SOFTWARE FAST**

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor of Philosophy
aan de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. V. Subramaniam,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Bètawetenschappen
op maandag 28 september 2020 om 15.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Taddeus Kroes

geboren te Amsterdam

promotor: prof.dr.ir. H.J. Bos
copromotor: dr. C. Giuffrida

“This will be a very short project, I promise.”

Cristiano Giuffrida, in 2016, 2017, 2018 and 2019.

Acknowledgements

Even though doing a Ph.D. is often a lonely journey, at the end of it I have many people to thank for being there along the way.

First, I want to thank Herbert Bos for being an amazing supervisor. You gave me complete freedom in pursuing my own projects, and empowered me to conduct research with pride and integrity. Even when I lost all motivation after a sea of negative reviews for some projects, you convinced me the work I did was “just super cool”. I am grateful to have been exposed to that energy.

I am equally grateful for the energy of my copromotor, Cristiano Giuffrida. You are the most creative researcher I could imagine, always coming up with cool new ideas. I will miss our conversations that, while they always took at least twice the allocated time, never failed to leave me knowing exactly what to do next.

Next, I sincerely thank the members of my thesis committee: Stefan Bunthaler, Cristian Cadar, Thorsten Holz, Alex Iosup, and Asia Slowinska. Thank you all for dedicating your time to reviewing my work, and for helping me improve it.

During my research, I was fortunate enough to collaborate with some great people. In fact, I share first authorship on several chapters of this dissertation. Special thanks go to Koen Koning. We started off working on Delta Pointers together, but also after that you helped me out countless times in work you were not even involved in. But more importantly, you have been a good friend all throughout. Erik, thank you for involving me in the Type-After-Type paper and for making it a notably smooth co-working experience. Anil and Joseph, thank you for continuing my work on BinRec all the way up in Irvine, California. Many thanks to all the students and postdocs there who helped publish the work after such a long struggle.

In addition to the colleagues I collaborated with directly, I will never forget working alongside so many smart people at VUsec. Thanks to my roomie Lucian for all our conversations and friendship. To István, for teaching me how binaries work. To Alyssa for being a smart and sensible sounding board. To Kaveh for the laughs. To all the others for some great borrels and chats: Andrea, Andrei (both of them), Angelos, Ben, Brian, Dennis, Elia, Emanuele, Enes, Erik B., Hany, Kousthuba, Manolis, Manuel (get Vim!), Marco, Marius (thanks for the thesis pass!), Michael, Natalie, Pietro, Radhesh, Sanjay, Sebastian, Stephan, and Vic-

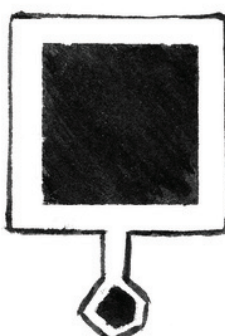
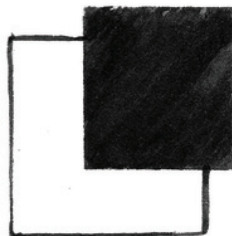
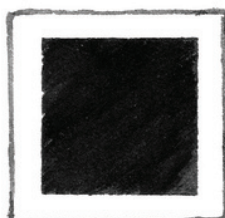
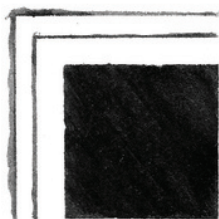
tor. Thank you all! And of course, thank you Caroline and Mojca for ever so swiftly taking care of the administrative “rompslomp”.

In the last year of my Ph.D., I went on an internship at Google in New York. Although that has little to do with this thesis directly, I view my experience there as part of my Ph.D. journey and would like to take this opportunity to thank the people involved in making it an awesome one. László and Wei, thank you for supervising me and giving me so much freedom in my work. Stefan, I am happy to have gained you as a friend, and grateful for your professional guidance. Markus, thanks for showing me all the cool lunch spots! Franjo, thank you for showing me that managers can actually be a great influence.

Besides all the people I met at work, there are just as many people to thank for keeping me sane in my personal life. Rowan, buddy, thanks for being there during the lows and the highs. For all the long gaming sessions in which I could vent. For the meaningful talks, but especially the meaningless ones. Raphael, you are an excellent person and friend. I cherish our conversations and the way they have helped me grow. Stijn, I enjoyed our frequent Friday night beers & laughs about academia. Ger and Renate, thanks for all the board game nights! To Richard, Sander, Jayke, Gijs, Rik, and all my other nerd friends: I’m looking forward to our next Corona-free borrel! Simon, our bike rides have been a great way to convert all kinds of energy into a positive kind. Let’s keep them coming. To my parents and numerous siblings: thank you for always supporting my choices.

Mei-Li, words cannot describe how lucky and grateful I feel to have you in my life. You have made me a more complete person. Thank you for supporting me. Thank you for being you. I also love that we got to work together sometimes. Or, rather, you working hard to make computer science look as cool as it can be. You made everything from conference posters, to course mascots, to the beautiful illustrations in this thesis(!) Thank you for letting me share my work with you in that way, it means the world to me.

Taddeüs Kroes
Amsterdam, The Netherlands, August 2020



Contents

Acknowledgements	vii
Contents	xi
List of Figures	xiv
List of Tables	xvi
List of Listings	xvii
Publications	xix
1 Introduction	3
2 Delta Pointers: Buffer Overflow Checks Without the Checks	11
2.1 Introduction	12
2.2 Background	13
2.3 Threat model	15
2.4 Delta Pointers	15
2.5 Pointer tagging	19
2.5.1 C pointer operations	20
2.5.2 Compiler support	21
2.5.3 Coverage considerations	24
2.5.4 Performance considerations	25
2.6 Implementation	26
2.6.1 Address space reduction	26
2.6.2 Instrumentation	26
2.6.3 Coverage	27
2.6.4 Optimization	28
2.7 Evaluation	29
2.7.1 Runtime performance	30

2.7.2	Security	33
2.8	Discussion	35
2.9	Related work	38
2.10	Conclusion	39
3	RedPool: Slabifying Redzones for Efficient Spatial Memory Safety	45
3.1	Introduction	46
3.2	Background	47
3.3	Redzoning for security	48
3.4	Threat model	49
3.5	RedPool	49
3.5.1	Slabification	51
3.5.2	Fast checks using guard values	54
3.5.3	Redzone initialization and reuse	55
3.5.4	Check elimination and merging	56
3.5.5	Shadow memory	57
3.6	Implementation	58
3.6.1	Slabification	59
3.6.2	Inserting redzone checks	60
3.6.3	Page fault handling for redzone reuse	60
3.6.4	Uninstrumented libraries	60
3.7	Evaluation	61
3.7.1	Performance	61
3.7.2	Comparison of redzone-based mitigations	65
3.7.3	Comparison with pointer-based bounds checkers	66
3.7.4	Redzone characterization	68
3.7.5	Web servers	69
3.7.6	Security	71
3.8	Related work	73
3.9	Discussion and future work	74
3.10	Conclusion	75
4	Type-After-Type: Practical and Complete Type-Safe Memory Reuse	77
4.1	Introduction	78
4.2	Background	80
4.2.1	Use-after-free	80
4.2.2	Uninitialized reads	80
4.2.3	Type safety	81
4.3	Threat model	81

4.4	Overview	81
4.5	Heap	83
4.5.1	Typed memory allocations	83
4.5.2	Wrapper detection and inlining	85
4.6	Stack	86
4.6.1	Guaranteed initialization on the safe stack	86
4.6.2	Typed unsafe stacks	87
4.7	Implementation	88
4.8	Evaluation	88
4.8.1	Security	89
4.8.2	Type detection	90
4.8.3	Wrapper detection and inlining	91
4.8.4	Memory overhead	92
4.8.5	Runtime overhead	92
4.8.6	Firefox case study	94
4.9	Limitations	96
4.10	Related work	97
4.11	Conclusion	100
5	BinRec: Dynamic Binary Lifting and Recompilation	103
5.1	Introduction	104
5.2	Current limitations in binary lifting	106
5.2.1	(C1) Code vs data, and reference ambiguity	106
5.2.2	(C2) Indirect control flow	106
5.2.3	(C3) External entry points	107
5.2.4	(C4) Ill-formed code	108
5.2.5	(C5) Obfuscation	108
5.3	Design	108
5.3.1	Key considerations for dynamic lifting	109
5.3.2	Dynamic lifting engine	111
5.3.3	Canonicalization	112
5.3.4	Lowering	116
5.3.5	Control flow miss handling	116
5.4	Implementation	117
5.4.1	Parallel tracing	118
5.4.2	Optimization	118
5.5	Evaluation	119
5.5.1	Comparison with static lifters	119
5.5.2	Performance	121

5.5.3	Code coverage	122
5.5.4	Lifting time	123
5.6	Applications	124
5.6.1	Control-flow hijacking mitigation	125
5.6.2	AddressSanitizer	127
5.6.3	SafeStack	127
5.6.4	Virtualization-deobfuscation	128
5.7	Limitations	128
5.8	Related work	130
5.9	Conclusion	131
6	Conclusions	133
	References	139
	Contributions to papers	153
	Summary	155
	Samenvatting	157

List of Figures

2.1	Encoding of Delta Pointers.	17
2.2	Delta tag updates on pointer arithmetic.	17
2.3	Delta tag masking on memory access.	17
2.4	Runtime overhead of Delta Pointers on SPEC CPU2006.	30
2.5	Breakdown of runtime overhead.	32
2.6	Overhead of Nginx web server for Delta Pointers.	33
2.7	Delta Pointers presentation poster for EuroSys'18	42
3.1	RedPool memory organization.	50
3.2	An example of slabification.	52
3.3	Example of fast checks emitted for a stack object.	55
3.4	Runtime and memory overhead of RedPool on SPEC CPU2006.	62
3.5	Breakdown of runtime overhead without redzone reuse.	63
3.6	Breakdown of runtime overhead with redzone reuse.	64
3.7	Runtime overhead of RedPool, RedShadow, and ASan.	67
3.8	Memory overhead of RedPool, RedShadow, and ASan.	67
3.9	Byte values at dereferenced addresses in SPEC CPU2006.	69
3.10	Web server throughput with increasing client connections.	70
3.11	Distribution of struct type sizes in SPEC CPU2006.	72
4.1	Overview of Type-After-Type framework.	82
4.2	Memory overhead of Type-After-Type on SPEC CPU2006.	92
4.3	Runtime overhead of Type-After-Type on SPEC CPU2006.	93
5.1	Steps of binary recovery in BinRec.	110
5.2	Address space layout of recovered code.	115
5.3	Impact of optimizations on recovered code.	122
5.4	Coverage of recovered code with respect to original binaries.	123
5.5	Incremental lifting progression of <i>bzip2</i>	124
5.6	Virtualization-deobfuscation approach based on BinRec.	129
5.7	Deobfuscation example of the <i>fib</i> program.	129

List of Tables

2.1	Comparison of overflow checkers.	37
2.2	Detailed overhead numbers of Delta Pointers on SPEC CPU2006. . . .	40
2.3	Raw runtime numbers of Delta Pointers on SPEC CPU2006.	41
3.1	Web server overhead at saturation.	70
4.1	Vulnerabilities thwarted by Type-After-Type.	89
4.2	Runtime overhead of Type-After-Type on Firefox	95
5.1	Overhead of binaries recovered by BinRec.	121
5.2	Lifting performance of BinRec, S ² E, and McSema.	125
5.3	Allowed control flow targets in BinRec and BinCFL.	126
5.4	Code size reduction by deobfuscation.	129



List of Listings

- 2.1 Delta Pointers instrumentation. 19
- 3.1 RedPool instrumentation on a stack object. 53
- 3.2 Check merging in RedPool. 57
- 5.1 Excerpt of decompress.c in libjpeg. 120

Publications

This dissertation includes several research papers, as appeared in the following conference proceedings. The text differs from the published versions in minor editorial changes made to improve readability, and added clarifications to address comments by the committee:

Taddeus Kroes¹, Koen Koning¹, Erik van der Kouwe, Herbert Bos, and Cristiano Giuffrida. **Delta Pointers: Buffer Overflow Checks Without the Checks.** In *Proceedings of the Thirteenth EuroSys Conference (EuroSys '18)*, page 22. April 23-26, 2018, Porto, Portugal.
[Appears in Chapter 2]

Taddeus Kroes, Chris Ouwehand, Cristiano Giuffrida, and Herbert Bos. **RedPool: Slabifying Redzones for Efficient Spatial Memory Safety.** *Under review*
[Appears in Chapter 3]

Erik van der Kouwe, Taddeus Kroes, Chris Ouwehand, Herbert Bos, and Cristiano Giuffrida. **Type-After-Type: Practical and Complete Type-Safe Memory Reuse.** In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*, pages 17–27. December 3-7, 2018, San Juan, Puerto Rico.
[Appears in Chapter 4]

Anil Altinay¹, Joseph Nash¹, Taddeus Kroes¹, Prabhu Rajasekaran, Dixin Zhou, Adrian Dabrowski, David Gens, Yeoul Na, Stijn Volckaert, Cristiano Giuffrida, Herbert Bos, and Michael Franz. **BinRec: Dynamic Binary Lifting and Recompile.** In *Proceedings of the Fifteenth EuroSys Conference (EuroSys '20)*, pages 1–16. April 27-30, 2020, online virtual conference.
[Appears in Chapter 5]

¹Equal contribution shared first authors.

Related publications not included in the dissertation are listed in the following:

Taddeus Kroes¹, Koen Koning¹, Cristiano Giuffrida, Herbert Bos, and Erik van der Kouwe. **Fast and Generic Metadata Management with Mid-Fat Pointers**. In *Proceedings of the 10th European Workshop on Systems Security (EuroSec '17)*. April 23, 2017, Belgrade, Serbia.

Taddeus Kroes¹, Anil Altinay¹, Joseph Nash¹, Yeoul Na, Stijn Volckaert, Herbert Bos, Michael Franz, and Cristiano Giuffrida. **BinRec: Attack Surface Reduction Through Dynamic Binary Recovery**. In *Proceedings of the 2018 Workshop on Forming an Ecosystem Around Software Transformation (FEAST '18)*, pages 8–13. November 15, 2019, Toronto, Canada.

Publications not related to this dissertation, but published in refereed conferences are listed in the following:

Lucian Cojocar, Taddeus Kroes, and Herbert Bos. **JTR: A Binary Solution for Switch-Case Recovery**. In *Proceedings of the International Symposium on Engineering Secure Software and Systems 2017 (ESSoS '17)*. July 4–5, 2017, Bonn, Germany.

¹Equal contribution shared first authors.

